Energy, Mines and Resources Canada
Énergie, Mines et Ressources Canada

CANMET
Canada Centre for Mineral and Energy Technology
Centre canadien de la technologie des minéraux et de l'énergie

NEURAL NETWORKS AND THEIR APPLICATIONS IN MINERAL PROCESSING: PART 1. WHAT IS A NEURAL NETWORK?

J. Leung
Core Technologies and Services Laboratory

May 1992

This document is an unedited
and as report previously
for
reproduction. It does not
represent final expression of
the opinions of the Canada Centre
for Mineral and Energy Technology
(CANMET)

Ce document est un rapport
préparé à titre préliminaire
pour
diffusion et ne représente pas
exactement l'opinion du Centre canadien de
la technologie des minéraux et de
l'énergie (CANMET)

21 pp

Project: **30.50.99**
Computer-Aided Mineral Processing

MINERAL SCIENCES LABORATORIES
DIVISION REPORT MSL 92-20 (IR)

MICROMEDIA

Crown Copyrights Reserved

NEURAL NETWORKS AND THEIR APPLICATIONS IN MINERAL PROCESSING: PART 1. WHAT IS A NEURAL NETWORK?

by

J. Leung*

ABSTRACT

The objective of this review of neural network technology is to explore how this technology can be applied in the mineral processing industry. Although neural networks have existed for some time, they generally remain in the research laboratory with very few industrial examples reported. This report explains the nature of neural networks by first presenting the background of the technology; then describing the network architecture based on the Back-Propagation network; and finally introducing a few important network models mostly used by researchers. Knowledge of neural networks should demystify any illusions that might have existed and encourage researchers in the mineral processing industry to develop ways that neural networks can contribute in their process control and optimization.

Keywords: Neural networks, mineral processing, back-propagation neural network.

---

* Core Technologies and Services Laboratory, CANMET, Energy, Mines and Resources, 552 Booth Street, Ottawa, Ontario, K1A 0G1.

93-00501/

TABLE OF CONTENTS

	Page
ABSTRACT	i
INTRODUCTION	1
BACKGROUND	2
NETWORK ARCHITECTURE	2
Biological Inspiration	2
Processing Element	3
Network Layer	4
Learning Types	6
Learning Algorithms	7
Connection weights between hidden and output layers	7
Connection weights between input and hidden layers	9
Summary	10
NETWORK MODELS	11
ADALINE and MADALINE	12
Adaptive Resonance Theory Networks (ART)	12
Recirculation Networks	13
Back-Propagation Network	14
CONCLUSIONS	14
ACKNOWLEDGEMENT	15
REFERENCES	15
APPENDIX: GLOSSARY	17

FIGURES

No.	Page
1. A processing element	3
2. A simple neural network architecture	5

INTRODUCTION

The interest in neural networks, or connectionist models, is due mainly to the computational and information processing abilities of a fully connected network of neurons. Unlike expert systems where knowledge is made explicit in the form of rules, neural networks generate their own rules by learning from examples. Besides the ability to learn, they also have the capability of performing massive parallel processing, which is in contrast to conventional digital computers in which the instructions are executed sequentially. Many new technologies, such as software simulation in parallel process programming, hardware accelerators, silicon chips, optical processors, etc., have been developed to facilitate the parallel and distributed computing capability of neural networks. Neural networks, due to their characteristics, have the potential to solve very complicated nonlinear processes normally encountered in mineral processing. The technology has been identified as extremely useful in the following areas: language and signal processing; pattern classification; modelling and simulation; as well as process control and optimization. Although the technology has been the focus of attention for many disciplines in the past decade, it is only recently that neural networks have been considered for mineral processing applications by some research institutes.

This is the first part of the report that introduces neural networks and their applications in mineral processing. This part describes the nature of neural network technology. A brief background history and network architecture, based on the Back-Propagation Network, are presented along with a description of some common network models. A glossary is also presented in the appendix for quick reference. The second part of the report will compare neural networks with other existing technologies such as statistical methods and expert systems. Finally possible applications of neural networks will be discussed followed by some examples using laboratory or industrial plant data for illustration purposes.

BACKGROUND

Since the paper on the topic of nervous activity was published by McCulloch and Pitts (1) in 1943, it had inspired researchers from diverse fields, from neurobiologists to psychologists and from physicists to computer engineers, in attempting to develop "artificial brains" that could perform some of the functions that were characteristic of human brains. This inspiration led directly and indirectly to the development of digital computers and artificial intelligence which then evolved into expert systems and neural networks.

The concept of neural computing first emerged by Rosenblatt in 1958 in his development of a "perceptron" (2). A perceptron is a pattern classification system that can learn to recognize symbols. The system consists of three layers: the first layer, known as the "input layer" where each input buffer element receives one signal from a portion of the input pattern; the second layer, known as the "feature demon layer" where the elements are fully or randomly connected to the input layer; and the third layer, known as the "perceptron layer" where each element represents the feature of the input pattern that is to be identified with. This novelty spurred research in neural networks until 1969, when Minsky and Papert published a paper showing its limitations (3). The paper dampened most of the interest and research in this area. The credibility of neural networks was not regained until 1982 when Hopfield illustrated and modelled the capabilities of neuron operation in a fully connected network (4). Enthusiasm for neural networks resurged and as a result, today, many neural models have been developed for various applications. Advanced technologies such as very large scale integration (VLSI) (5), optical processors (6), silicon chips etc., are also under investigation in order to facilitate neural computing systems.

NETWORK ARCHITECTURE

Biological Inspiration

After years of research, scientists started to have a clearer picture of the overall organization of the brain. The brain consists of tens of billions of neurons densely interconnected in a

very small space. The axon (output path) of a neuron splits up and connects to dendrites (input paths) of other neurons through a junction referred to as a synapse. Each neuron acts like a simple microprocessing unit which receives and combines signals from as many as 10^4 other neurons of 100 different types. If the combined signal of the input paths is sufficiently strong, it activates the firing of the neuron and produces an output signal which may connect to as many as 10^4 other neurons. When the brain learns, the "strength" of the connections (or synaptic strength in biological terms) between the input and output paths changes in a way that is either excitatory or inhibitory. Increases in strength through reinforcement are called "learning" and decreases are called "forgetting". As many as 10^{12} connection strengths can be involved in a thinking process in about 5 milliseconds. It is these vast numbers of interconnected neurons with information processing ability that are the inspiration behind the architecture of neural networks.

Processing Element

A typical network consists of several layers of processing elements. The structure of a processing element, analogous to the biological neuron, is shown in Figure 1.

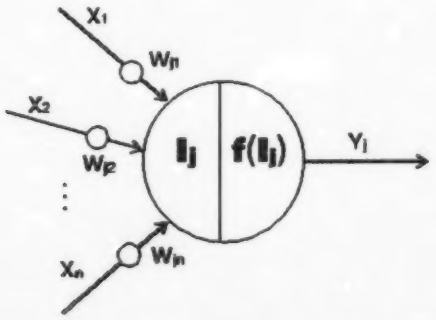

Fig. 1 A processing element

The j^{th} processing element has many input paths X_i and the weighted values $W_{ji}X_i$ are combined to form an internal activity level I_j. The combined input is then passed through a

transfer function f before going directly to an output path Y_j, which in turn may become the input to one or more processing elements. The relationship can be written as:

$$I_j = \sum_{i=1}^{I} W_{ji} X_i \tag{1}$$

$$Y_j = f(I_j) = f\left(\sum W_{ji} X_i \right) \tag{2}$$

The transfer function can either be a threshold function which only passes information if the combined activity level reaches a certain level, or it can be a linear or nonlinear function such as a sigmoid, sine or hyperbolic tangent function.

Network Layer

The power of neural networks lies in the way these simple processing elements are interconnected between successive layers. Typically, there are two layers with connections to the outside world: an input buffer where data is presented to the network and an output buffer which holds the response of the network corresponding to a given input data set. The layers between the input and output layers are called hidden layers (or intermediate layers). It depends on the complexity of the problem, but there is usually not more than one hidden layer in a network. For the type of problems usually encountered in mineral processing, a three-layer network is sufficient to represent the dynamic behaviour of the process. However, the number of elements used in the hidden layer depends on the application and is determined empirically.

Figure 2 shows a typical three-layer Back-propagation network, also known as feedforward network. Assuming X is the input vector in the input layer and H is the output vector in the hidden layer, as in Eqns. 1 and 2, the output H of element j in the hidden layer can be

written as:

$$H_j = f(I_j) = f\left(\sum_{i=1}^{I} W_{ji} X_i\right) \qquad 1 \le j \le J \tag{3}$$

where W_{ji} is the connection weights which are determined during a learning or training process, f is the transfer function, I_j is the internal activity level and J the total number of elements in the hidden layer. Similarly the output Y of element k in the output layer can be written as:

$$Y_k = f(I_k) = f\left(\sum_{j=1}^{J} W_{kj} H_j\right) \qquad 1 \le k \le K \tag{4}$$

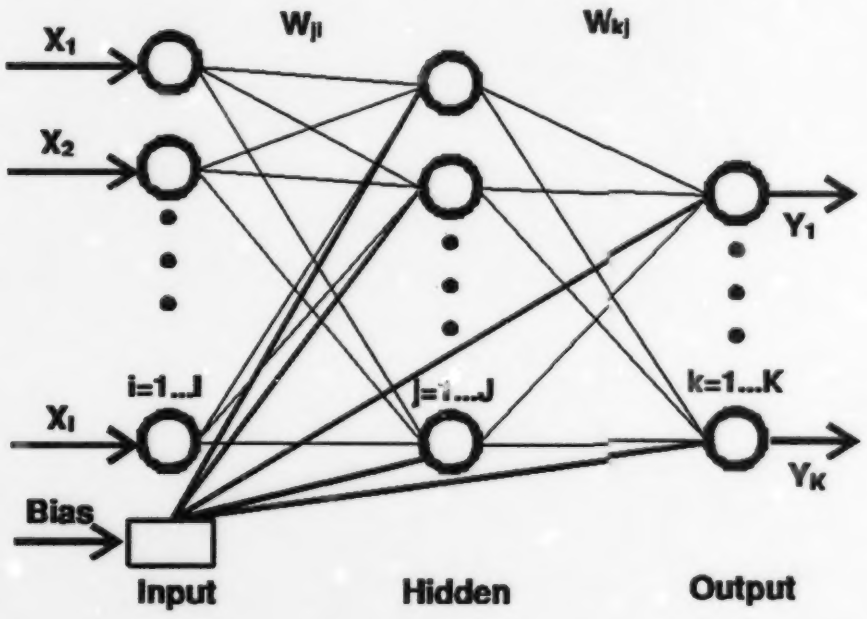

Fig. 2 A simple neural network architecture

Input and output values are usually scaled between 0 and 1 or -1 and 1 to avoid possible saturation caused by large values in the summations. Figure 2 also shows a bias element in the input layer. The bias, whose value is always fixed to 1, works like a threshold or "ground" in an electrical circuit. The bias, which connects to each processing element in the hidden and output layers, adds a constant value to each weighted sum and its connection weight is learned in the same manner as the other weights are learned.

Learning Types

Learning is the most important process in the network operation. Neural networks are trained with examples. If a training set of input-output vectors are given, learning is simply to adapt or modify the connection weights in such a way that the underlying structure of the input-output relationship is captured. When a desired output is provided for a given input, the learning is referred to as "supervised learning". For example, a Back-propagation network uses "supervised learning" for training. "Unsupervised learning" means the network is left to self-organize the data without any desired output provided for a given input. One example of the network that applies unsupervised learning is the Adaptive Resonance Theory (ART) networks which will be described in the Section Network Models.

If, for all training examples, the desired output vector is transformed from the input vector, e.g., in the applications of noise filtering or signal encoding and compression, the trained network is called "auto-associative". Whereas, if the desired output is different from the input, e.g., in the applications of pattern classification or system modelling, the trained network is called "hetero-associative".

Whatever kind of learning type is used, a learning process is an iterative procedure that may require showing a network a given set of data many thousands of times until the network is fully trained. By analogy, the letters of alphabets must be shown to a child many times before he/she is able to recognize them.

Learning Algorithms

Using a three-layer neural network as an example and the nomenclature of Figure 2, the learning algorithms of a typical Back-propagation network are described in this section. The network uses a gradient descent algorithm to minimize an error function, i.e., the sum of the squares of the errors between the calculated outputs Y and desired outputs D in the output layer. For each particular input-output set (X,D) presented to the network, the error function E is defined as:

$$E = \frac{1}{2}\sum_{k=1}^{K}(Y_k - D_k)^2 \tag{5}$$

Connection weights between hidden and output layers

Given an initial set of weights W_{kj} between the hidden and output layers (usually set to small random values between -0.1 and 0.1), one can minimize the error function by repeatedly changing each weight by an amount proportional to $\partial E/\partial W_{kj}$, i.e.:

$$\Delta W_{kj} = -\epsilon\frac{\partial E}{\partial W_{kj}} \tag{6}$$

where ϵ is a learning coefficient. In other words, each weight is changed according to the size and direction of negative gradient on the error surface. Choosing a small ϵ and the number of updates going to infinity, this learning procedure will find a set of weights that gives the least squared error. Substituting with Eqns. 4 and 5, the partial derivative of Eq. 6, using the chain rule, becomes:

$$\frac{\partial E}{\partial W_{kj}} = \frac{\partial E}{\partial I_k}\frac{\partial I_k}{\partial W_{kj}} = \frac{\partial E}{\partial I_k}H_j \tag{7}$$

where:

$$\frac{\partial E}{\partial I_k} = \frac{\partial E}{\partial Y_k}\frac{\partial Y_k}{\partial I_k} \tag{8}$$

$$= f'(I_k)*(Y_k - D_k)$$

Combining the above equations, the error gradient $\partial E/\partial W_{kj}$ used to correct the weights going from the j^{th} element in the hidden layer to the k^{th} element in the output layer is:

$$\frac{\partial E}{\partial W_{kj}} = f'\left(\sum_{j=1}^{J} W_{kj}H_j\right)*(Y_k - D_k)H_j \tag{9}$$

where $f'[...]$ represents the derivative of the transfer function and H_j the output of the transfer function of internal activity level in element j (see Eq. 3). Normally nonlinear transfer functions like sigmoid or hyperbolic tangent functions are used. For sigmoid, the function and derivative are defined as:

$$f(x) = \frac{1}{(1+e^{-x})} \quad ; \quad f'(x) = f(x)*(1-f(x)) \tag{10}$$

and for hyperbolic tangent, the function and derivative are:

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad ; \quad f'(x) = (1+f(x))*(1-f(x)) \tag{11}$$

Both functions are quite similar in shape (s-shaped), except that the output range for hyperbolic tangent is -1 to +1 and for sigmoid is 0 to 1. Since the output of the transfer function H_j (Eq.3) is used as a multiplier in the weight update equation (Eq. 9), a range of 0 to 1 means a smaller multiplier when the summation is a low value and a higher multiplier for higher summations. This could lead to a bias of learning higher desired outputs i.e.,

approaching 1. The hyperbolic tangent gives equal weight to low and high end values and this is why it is preferable.

Connection weights between input and hidden layers

Similarly, to correct the weights, going from the i^{th} element in the input layer to the j^{th} element in the hidden layer, that have effect on the error, one can write the delta weights as:

$$\Delta W_{ji} = -\epsilon \frac{\partial E}{\partial W_{ji}} \tag{12}$$

and the error gradient as:

$$\frac{\partial E}{\partial W_{ji}} = \frac{\partial E}{\partial I_j}\frac{\partial I_j}{\partial W_{ji}} = \sum_{k=1}^{K} \frac{\partial E}{\partial I_k}\frac{\partial I_k}{\partial I_j} X_i \tag{13}$$

where the Σ term means that the j^{th} element in the hidden layer receives errors propagated back from all K elements in the output layer. The way that the network back propagates the errors thus is the reason for the name "Back-propagation network". The second partial derivative in Eq. 13 can further be expressed as:

$$\frac{\partial I_k}{\partial I_j} = \frac{\partial I_k}{\partial H_j}\frac{\partial H_j}{\partial I_j} \tag{14}$$

Replacing all the partial derivatives in Eqns. 13 and 14 by Eqns. 3,4 and 8, the error gradient $\partial E/\partial W_{kj}$ of Eq. 12 becomes:

$$\frac{\partial E}{\partial W_{ji}} = f'(I_j) * \left[\sum_{k=1}^{K} f'(I_k)(Y_k - D_k) W_{kj}\right] X_i \tag{15}$$

where I_j, I_k are the internal activity levels of the processing elements respectively in the hidden and output layers.

Summary

Therefore, the connection weights in the network are updated by adding the delta weights (in Eqns. 6 and 12 with the error gradients respectively replaced by Eqns. 9 and 15) to the corresponding weights of the previous iteration. Assuming m is an iteration counter, each weight is modified for the next iteration by an amount defined for connections between hidden and output layers as:

$$\begin{aligned} \Delta W_{kj}^{m} &= W_{kj}^{m} - W_{kj}^{m-1} \\ &= -\epsilon\, f'\left(\sum_{j=1}^{J} W_{kj} H_j\right) * (Y_k - D_k) H_j \end{aligned} \tag{16}$$

For connections between input and hidden layers, the amount is defined as:

$$\begin{aligned} \Delta W_{ji}^{m} &= W_{ji}^{m} - W_{ji}^{m-1} \\ &= -\epsilon f'\left(\sum_{i=1}^{I} W_{ji} X_i\right) * \left\{\sum_{k=1}^{K} f'\left(\sum_{j=1}^{J} W_{kj} H_j\right)(Y_k - D_k)\, W_{kj}^{m-1}\right\} * X_i \end{aligned} \tag{17}$$

where ϵ is the learning coefficient which determines the step size during gradient descent minimization.

In summary, the main mechanism in a Back-propagation network is to forward propagate the inputs through the layers to the output layer; to determine the error in the output layer and to propagate the errors back through the network from the output layer to the input layer. The aim of the learning process is to minimize the global error E as defined in Eq. 5 by modifying the weights in the network. The weights are updated by adding the delta weights (as defined in Eqns. 16 and 17 for different connections) to the corresponding previous

weights. The iteration process continues for every input-output pair of data presented to the network until the root mean squares of the output error is less than the convergence criterion.

There are some enhancements (7) that have been developed to improve the convergence speed or performance of this basic algorithm. For example, adding a momentum term in Eqns. 6 and 12 can speed up learning in directions that have a small but constant gradient. For example, the delta weight in Eq. 6 becomes:

$$\Delta W_{kj} = -\epsilon \frac{\partial E}{\partial W_{kj}} + \beta \Delta W_{kj}^{*} \tag{18}$$

where β is the momentum rate and ΔW_{kj}^{*} the last weight change. Algorithms like Delta-Bar-Delta (DBD) or Extended Delta-Bar-Delta (EDBD) are the extensions that allow the learning rate or the momentum rate to vary with time. The EDBD algorithm even permits each connection in the network to have its own rates. Another way to improve the convergence speed is to use second order derivatives to produce a more accurate estimate of the weight change. Convergence performance can be improved by updating the weights cumulatively. This is done by accumulating the delta weights until the completed set or subset of training examples is presented. This will ensure that the overall error function instead of the individual error function is reduced.

NETWORK MODELS

Many types of neural nets have been developed over the years for different types of applications. Models such as ADALINE, MADALINE, Feedforward, Boltzmann machine, Brain-State-in-a-Box, Counter-Propagation, Directed Random Search, Function Link, Recirculation, Self Organizing Map, etc., are some of them. The reference to the models can be found in the bibliographies (8, 9). Some of these important models will be briefly described with various applications in this section as an introduction to the subject.

ADALINE and MADALINE

Besides the "perceptron" one of the earlier models that have been designed is called ADALINE (from ADAptive LInear NEuron) by Widrow (10) in 1959. Like the "perceptron", an ADALINE network consists of three layers. But the output layer in ADALINE is a set of buffers and has the same number of processing elements as the middle layer. The network is also capable of classifying linearly separable patterns. Some multi-layer extensions to the ADALINE paradigm provide a better adaptive technique for separating the input space. Widrow later used an adaptive network to eliminate echoes on phone lines (11) and this was the first time a neural computing system was applied to a major real-world problem. A MADALINE (multiple ADALINES) network uses several ADALINE units in parallel with a single output unit. The output unit (known as a MADALINE unit) takes a majority vote on the inputs from the ADALINE units in the middle layer. In other words, the binary outcome (either -1 or +1) of the output is decided by the majority of the inputs which are also bi-state -1 and +1. Therefore, the MADALINE can only be used to discriminate between two classes. To discriminate between more than two classes, many independent MADALINES can be used, one for each pair of classes; then the final classification is the class that is output more times than any of the others. Both the ADALINE and MADALINE are used for a variety of applications including speech recognition, character recognition, weather prediction and adaptive control.

Adaptive Resonance Theory Networks (ART)

In 1976, Grossberg (12) built upon the work of his predecessors and developed an Adaptive Resonance Theory network (ART), based on the ideas that the brain spontaneously organizes itself into recognition states. The network can classify and recognize input patterns without any supervision from a "teacher", i.e., no desired response for a given input stimulus is provided in the network. This learning technique is known as "unsupervised learning". The

network categorizes the input pattern through its self-organizing ability (13, 14). An ART network consists of two interconnected layers of neurons. Each pixel of the input image or pattern is entered through a neuron in the input layer and the input pattern is categorized by one neuron in the output layer. During the learning process, the exchange of bottom-up and top-down information between the two layers leads to a resonance in neural activity. Critical features in the input layer are reinforced and have the greatest activity. The neurons in the output layer compete with one another yielding a "winner-take-all" network. Only the most active neuron remains active, all other neuron becoming inactive. However, there are some limitations when ART is used to categorize characters, especially the complicated ones such as Chinese calligraphy. Erroneous classification will result if the input patterns are "noisy". Investigations for improvements are underway by Fukushima who has developed an advanced neural computing system called "neocognitron" which can recognize handwritings, regardless of their orientation or distortion (15).

Recirculation Networks

Geoffrey Hinton and James McClelland (16) introduced this network in 1988. Instead of finding the dynamic relationship for the input-output data set, the network uses an unsupervised learning technique similarly to the ART network. A typical recirculation network consists of two layers: a visible layer and a hidden layer which are fully interconnected to each other in both directions. The purpose of the learning rule is to construct in the hidden layer an internal representation of the data presented at the visible layer. During the learning process, the data vector is first presented at the visible layer, then filtered to the hidden layer, recirculated back to the visible layer and finally passed for a second time to the hidden layer. Learning occurs after the second pass through the network and seeks to reduce the difference between the inputs of the visible layer at the first and second passes as well as between the outputs of the hidden layer at the first and second passes. An important application of this is to compress the input data by using fewer processing elements in the hidden layer such that the hidden representation can be considered as a compressed version of the visible representation. Therefore it is mainly used in data

compression and noise filtering. An example of noise filtering using this network model will be given in the second part of the report.

Back-Propagation Network

Of all the neural nets that have been developed, this is probably the one used the most in various applications. Rumelhart (17) is one of the researchers credited with its development that started in 1982. This network is also known as a feedforward neural network. Information is passed from input buffer to the output buffer in one direction without any feedback connections. The reasons why Back-propagation network are so popular are: 1) the algorithm is based on sound mathematical principles; 2) it provides a method to adjust connection weights in a network with many layers; 3) most importantly, it has produced a number of successful applications such as sonar target classification (18), machine vision inspection (19) and system modelling and prediction (20). This network offers more potential than other models for the type of problems mostly encountered in system identification and process control, and this is the reason why it was chosen.

CONCLUSIONS

A brief description regarding the background, the architecture of Back-propagation networks and the network model types is presented here as an introduction to this technology. Those, who are interested in more details, can always refer to numerous books and papers that have been published over the years. With the massive parallel and distributed multi-layered structure described in this report, neural networks especially the Back-propagation networks can be a useful tool to model highly nonlinear behaviour normally encountered in mineral processing. It appears that neural-based model control strategy can be used as an alternative method to control dynamic process.

ACKNOWLEDGEMENT

The author gratefully acknowledges E. TerHeijden and M. Bilodeau for reviewing the manuscript and CANMET management for making this project possible.

REFERENCES

1. McCulloch, W.S., Pitts, W., "A Logical Calculus of the Ideas Immanent in Nervous Activity", Bull. Math. Biophys., 5, 115-133, 1943.

2. Rosenblatt, F., "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", Psycholog. Rev., 65, 386-408, 1958.

3. Minsky, M., Papert, S., "Perceptrons", MIT Press, Cambridge, 1969.

4. Hopfield, J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proc. of the Natl. Acad. Sci., 79, 2554-2558, 1982.

5. Mead, C.A., "Analog VLSI and Neural Systems", Addison-Wesley, Reading, MA, 1989.

6. Hsu, K.Y. et al., "Holographic Implementation of a Fully Connected Neural Network", Proc IEEE Vol. 78, No. 10, pp. 1637-1645, Oct. 1990.

7. Samad, Tariq, "Back-Propagation Extensions", Honeywell SSDC Technical Report, 1000 Boone Ave. N., Golden Valley, MN 55427, 1989.

8. Klimasauskas, C., "The 1989 Neuro-Computing Bibliography", MIT Press, Cambridge, 1989.

9. Wasserman, P.D., Oetzel, R.M., "Neural Source, The Bibliographic Guide to Artificial Neural Networks", Van Nostrand Reinhold, New York, 1990.

10. Widrow, B., Hoff, M., "Adaptive Switching Circuits", IRE WESCON Conv. Record, Part 4, 96-104, 1960.

11. Widrow, B., Glover, J.R., McCool, et al., "Adaptive Noise Cancelling: Principles and Applications", Proc. of the IEEE, Vol. 63, No. 12, December, 1975.

13. Carpenter, G.A., Grossberg, S., "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine", Computer Vision, Graphics and Image Processing 37, 54-115, 1987.

14. Carpenter, G.A., Grossberg, S., "ART2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns", Applied Optics, 4919-4930, 1987.

15. Fukushima, K., "Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition", Neural Networks, Vol. 1, 119-130, 1988.

16. Hinton, G.E., McClelland, J.L., "Learning Representations by Recirculation", Proceedings of IEEE Conference on Neural Information Processing Systems, Nov. 1988.

17. Rumelhart, D.E., Hinton, G.E., William, R.G., "Learning Internal Representations by Back-Propagating Errors", Nature, 323, 533-536, 1986.

18. Gorman, R.P. and Sejnowski, T.J., "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets", Neural Networks, 1, pp. 75-89, 1988.

19. Glover, David E., "A Hybrid Optical Fourier/Electronic Neurocomputer Machine Vision Inspection System", Proc. Vision '88 Conference, sponsored by SME/MVA.

20. Lapedes, A. and Farber, R., "Non-Linear Signal Processing using Neural Networks: Prediction and System Modelling", Los Alamos National Laboratory report LA-UR-87-2662, 1987.

APPENDIX: GLOSSARY

ADAPTIVE RESONANCE THEORY (ART):
A theory which address issues in the stable self-organization of networks for pattern classification without an external teacher.

AUTO-ASSOCIATIVE MEMORY:
A memory which is designed to transform an input pattern to itself. If the input pattern is noisy, degraded or incomplete the memory will still recall the undergraded pattern.

AXON:
The part of a neuron which is single and long, terminating in short branches relatively far from the nucleus and through which an energy signal is passed to another neuron.

BACK PROPAGATION:
A learning scheme by which a multi-layer network is organized for pattern recognition or classification utilizing an external teacher.

COMPETITION:
A situation where two neurons compete in that if one of them becomes more active the other becomes less active.

CONTROL STRATEGY:
Specifies the order in which network states are updated for digital implementation of complicated learning and recall functions.

DELTA RULE LEARNING:
A type of learning where weights are modified to reduce the difference between the desired output and the actual output of a processing element.

DENDRITES:
The part of neuron which resembles the branch of a tree, terminating near the nucleus and through which input signals are coming in from another neuron.

DISTRIBUTED ASSOCIATIVE MEMORY:
The type of memory prevalent in neural computing systems in which data is distributed throughout many weighted connections between processing elements, and in which outputs complete information on input of partial information.

FAULT TOLERANCE:

A property of neural computing systems that allows the system to function and gradually degrade when a small number of processing elements are destroyed or disabled.

FEEDFORWARD NETWORK:

A network in which information flow is all in one direction. In such networks there are no feedback loops from a processing element to a previous one.

GENERALIZATION:

The ability of a neural computing system to generalize from the input/output examples it was trained on to produce a sensible output to a previously unseen input.

HETERO-ASSOCIATIVE MEMORY:

A type of memory in which the pattern on recall from the memory is purposely different from the input pattern.

LAYER:

The main architectural component of a neural network consisting of a number of processing elements of equal functionality and occupying a position in the network corresponding to a particular stage of processing.

LEARNING:

Self-adaptation at the processing element level. Weighted connections between processing elements are adjusted to achieve specific results, eliminating the need for writing a specific algorithm for each problem.

LEARNING SCHEDULE:

A schedule which specifies how parameters associated with learning change over the course of training a network.

NETWORK:

A mathematical model of a computing system.

NEURAL COMPUTING:

The fast growing field of computing technology inspired by studies of the brain. Neural computing, from its origins, is ideally adapted to doing pattern matching, pattern rercognition and control function synthesis.

NEURON:

A nerve cell in the physiological nervous system.

PARALLEL PROCESSING:

A form of computing in which many computations are being processed simultaneously. One of the unique features of neural computing is that it provides an